**7 ЛАБОРАТОРНАЯ РАБОТА №7**

**РАБОТА В JAVA С ПОЧТОВЫМИ СЕРВИСАМИ**

**Цель работы**: изучить пакет JavaMail, позволяющий создавать почтовые приложения и осуществлять обработку электронной почты.

**7.1 Краткие теоретические сведения**

Java Mail – набор классов, которые обеспечивают платформенно и протокольно независимый доступ к сервисам электронной почты и позволяют создавать полнофункциональные почтовые приложения.

JavaMail разработан для облегчения возможности обработки электронной почты простыми приложениями. Пакет включает классы, которые реализуют общие почтовые функции и протоколы.

JavaMail API поддерживает различные реализации почтовых систем с различными форматами сообщений, с использованием различных транспортных протоколов. Обеспечивает набор базовых классов и интерфейсов, которые определяют API для приложений клиентов. Многим простым приложениям необходимо только взаимодействовать с системой передачи сообщений через эти базовые классы и интерфейсы. Пакет JavaMail входит в состав J2EE.

Абстрактный класс, который определяет набор атрибутов и содержание для почтового сообщения. Атрибуты класса Message определяют информацию адресации и определяют структуру содержания. Содержание представлено как объект DataHandler, который служит обверткой вокруг фактических данных. Класс Message реализует интерфейс Part. Интерфейс Part определяет атрибуты, которые требуются для определения и форматирования содержания данных, которые несет объект Message, и связывает с помощью интерфейса с почтовой системой. Класс Message добавляет атрибуты From, To, Subject, Reply-To и другие, необходимые для маршрутизации сообщения через транспортную систему сообщений. JavaMail содержит подклассы Message, которые поддерживают различные реализации.JavaMail API также поддерживает многослойные объекты Message, где каждый Bodypart определяет его собственный набор атрибутов и содержания.

Сообщения сохраняются в объектах Folder. Объект Folder может содержать подпапки так же как и сообщения, таким образом, обеспечивая древовидную структуру папки. В классе Folder объявлены методы, которые позволяют производить выборку, добавление в конец, копирование и удаление сообщений. Объект Folder может также посылать события компонентам, зарегистрированным как слушатели.

Класс Store определяет базу данных, которая скрепляет иерархию папки с ее сообщениями. Класс Store также определяетпротокол доступа, который обращается к папкам и получает сообщения, сохраненные в папках. Класс Store также реализует методы установления подключения с базой данных, выбора папки и закрытия подключение. Классы, реализующие протоколы доступа к сообщениям (IMAP4, POP3, и т.д.) начинаются, наследуясь от класса Store. Пользователь типично начинает сессию с почтовой системой, соединяясь с конкретной реализацией Store.

Пользователь создает новое сообщение путем создания экземпляра подкласса Message. Осуществляет установку атрибутов, таких как адрес получателя и тема и добавляет содержание в объект Message. После этого он посылает сообщение посредством вызова метода Transport.send().

Transport моделирует транспортного агента, который пересылает сообщение списку получателей. Вызов метода Transport.send() с объектом Message определяет соответствующий механизм передачи, основанный на его адресах назначения.

**7.2 Тексты разработанных классов**

package ua.cn.stu.lab7;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Properties;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JScrollPane;

import javax.swing.JTabbedPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import java.awt.Font;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.SwingConstants;

import java.awt.Component;

import java.awt.Color;

public class MainGUI {

private JFrame frame;

private static JTextArea textNewMessage;

private JPasswordField textPass;

private boolean working = false;

private JTextField textLogin;

private JButton btnStartcheck;

private MailChecker checkThread;

private MailSender sendThread;

private JTextField textTo;

private JTextField textTheme;

private Properties properties;

private JTextArea textBody;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException

| UnsupportedLookAndFeelException e1) {

e1.printStackTrace();

}

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

MainGUI window = new MainGUI();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

public MainGUI() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

frame = new JFrame();

frame.setBounds(100, 100, 500, 381);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);

JPanel panel = new JPanel();

tabbedPane.addTab("check email", null, panel, null);

btnStartcheck = new JButton("start check");

btnStartcheck.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

checkThread();

}

});

JScrollPane scrollPane = new JScrollPane();

textNewMessage = new JTextArea();

textNewMessage.setForeground(new Color(0, 0, 0));

textNewMessage.setEditable(false);

textNewMessage.setFont(new Font("Monospaced", Font.PLAIN, 12));

scrollPane.setViewportView(textNewMessage);

GroupLayout gl\_panel = new GroupLayout(panel);

gl\_panel.setHorizontalGroup(

gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel.createSequentialGroup()

.addGroup(gl\_panel.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel.createSequentialGroup().addContainerGap().addComponent(

scrollPane, GroupLayout.DEFAULT\_SIZE, 439, Short.MAX\_VALUE))

.addGroup(gl\_panel.createSequentialGroup().addGap(172).addComponent(btnStartcheck,

GroupLayout.PREFERRED\_SIZE, 107, GroupLayout.PREFERRED\_SIZE))).addContainerGap()));

gl\_panel.setVerticalGroup(gl\_panel.createParallelGroup(Alignment.LEADING).addGroup(Alignment.TRAILING,

gl\_panel.createSequentialGroup().addContainerGap()

.addComponent(scrollPane, GroupLayout.DEFAULT\_SIZE, 221, Short.MAX\_VALUE)

.addPreferredGap(ComponentPlacement.RELATED).addComponent(btnStartcheck)));

panel.setLayout(gl\_panel);

JPanel panel\_1 = new JPanel();

tabbedPane.addTab("SendEmail", null, panel\_1, null);

JButton btnSendEmail = new JButton("send email");

btnSendEmail.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

sendEmail();

}

});

JLabel lblTo = new JLabel("To");

textTo = new JTextField();

textTo.setColumns(10);

JLabel lblTheme = new JLabel("Theme");

textTheme = new JTextField();

textTheme.setColumns(10);

textBody = new JTextArea();

textBody.setFont(new Font("Monospaced", Font.PLAIN, 12));

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(25).addComponent(lblTo,

GroupLayout.PREFERRED\_SIZE, 17, GroupLayout.PREFERRED\_SIZE))

.addGroup(gl\_panel\_1.createSequentialGroup().addContainerGap().addComponent(lblTheme)))

.addGap(10)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addComponent(textTo, Alignment.TRAILING, GroupLayout.DEFAULT\_SIZE, 397,

Short.MAX\_VALUE)

.addComponent(textTheme, Alignment.TRAILING, GroupLayout.DEFAULT\_SIZE, 397,

Short.MAX\_VALUE)))

.addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addGap(20).addComponent(textBody,

GroupLayout.DEFAULT\_SIZE, 429, Short.MAX\_VALUE)))

.addContainerGap())

.addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addGap(191)

.addComponent(btnSendEmail, GroupLayout.PREFERRED\_SIZE, 89, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(179, Short.MAX\_VALUE)));

gl\_panel\_1.setVerticalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING).addGroup(gl\_panel\_1

.createSequentialGroup().addContainerGap()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE).addComponent(lblTo).addComponent(textTo,

GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE))

.addGap(13)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.BASELINE)

.addComponent(textTheme, GroupLayout.PREFERRED\_SIZE, 23, GroupLayout.PREFERRED\_SIZE)

.addComponent(lblTheme))

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textBody, GroupLayout.DEFAULT\_SIZE, 159, Short.MAX\_VALUE)

.addPreferredGap(ComponentPlacement.RELATED).addComponent(btnSendEmail)));

panel\_1.setLayout(gl\_panel\_1);

textPass = new JPasswordField("");

textLogin = new JTextField();

textLogin.setColumns(10);

JLabel lblLogin = new JLabel("Login");

JLabel lblPassword = new JLabel("Password");

JLabel lblgmailcom = new JLabel("@gmail.com");

GroupLayout groupLayout = new GroupLayout(frame.getContentPane());

groupLayout.setHorizontalGroup(groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(groupLayout.createSequentialGroup().addGap(10)

.addGroup(groupLayout

.createParallelGroup(

Alignment.LEADING)

.addGroup(groupLayout.createSequentialGroup()

.addComponent(lblLogin, GroupLayout.PREFERRED\_SIZE, 34,

GroupLayout.PREFERRED\_SIZE)

.addGap(10)

.addComponent(textLogin, GroupLayout.DEFAULT\_SIZE, 86, Short.MAX\_VALUE)

.addGap(10).addComponent(lblgmailcom).addGap(22).addComponent(lblPassword)

.addGap(10)

.addComponent(textPass, GroupLayout.DEFAULT\_SIZE, 108, Short.MAX\_VALUE)

.addGap(81))

.addComponent(tabbedPane, GroupLayout.DEFAULT\_SIZE, 464, Short.MAX\_VALUE)).addGap(10)));

groupLayout.setVerticalGroup(groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(groupLayout.createSequentialGroup().addGap(11)

.addGroup(groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(groupLayout.createSequentialGroup().addGap(3).addComponent(lblLogin))

.addComponent(textLogin, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGroup(groupLayout.createSequentialGroup().addGap(3).addComponent(lblgmailcom))

.addGroup(groupLayout.createSequentialGroup().addGap(3).addComponent(lblPassword))

.addComponent(textPass, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE))

.addGap(11).addComponent(tabbedPane, GroupLayout.DEFAULT\_SIZE, 289, Short.MAX\_VALUE)

.addGap(11)));

frame.getContentPane().setLayout(groupLayout);

}

public Properties getProperties() {

properties = new Properties();

properties.put("host", "smtp.gmail.com");

properties.put("mail.smtp.host", "smtp.gmail.com"); // SMTP Host

properties.put("mail.smtp.port", "587"); // TLS Port

properties.put("mail.smtp.auth", "true"); // enable authentication

properties.put("mail.smtp.starttls.enable", "true"); // enable STARTTLS

return properties;

}

private void sendEmail() {

if (sendThread == null)

sendThread = new MailSender();

String pass = new StringBuilder().append("").append(textPass.getPassword()).toString();

sendThread.setParam(textLogin.getText() + "@gmail.com", pass, textTo.getText());

sendThread.setProperties(getProperties());

sendThread.setBody(textBody.getText());

sendThread.setSubject(textTheme.getText());

sendThread.sendMessage();

}

private void checkThread() {

getProperties();

working = !working;

if (working) {

getTextArea().setText("");

btnStartcheck.setText("stop check");

checkThread = new MailChecker();

checkThread.setProperties(getProperties());

String pass = new StringBuilder().append("").append(textPass.getPassword()).toString();

checkThread.setParam(textLogin.getText() + "@gmail.com", pass, getTextArea());

try {

checkThread.start();

} catch (NullPointerException e) {

e.printStackTrace();

}

} else {

btnStartcheck.setText("start check");

if (checkThread.isAlive()) {

checkThread.setStatus(working);

}

}

}

public static JTextArea getTextArea() {

return textNewMessage;

}

}

package ua.cn.stu.lab7;

import java.io.UnsupportedEncodingException;

import java.util.Date;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import javax.swing.JOptionPane;

public class MailSender extends Thread {

private Properties props;

private String body;

private String subject;

private String from;

private String to;

private String password;

public MailSender() {

super();

}

public void sendMessage() {

props.put("mail.smtp.user", from);

props.put("mail.smtp.password", password);

Authenticator auth = new Authenticator() {

// override the getPasswordAuthentication method

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(from, password);

}

};

Session session = Session.getInstance(props, auth);

MimeMessage msg = new MimeMessage(session);

try {

msg.addHeader("Content-type", "text/HTML; charset=UTF-8");

msg.addHeader("format", "flowed");

msg.addHeader("Content-Transfer-Encoding", "8bit");

msg.setFrom(new InternetAddress(from, from));

msg.setReplyTo(InternetAddress.parse(to, false));

msg.setSubject(subject, "UTF-8");

msg.setText(body, "UTF-8");

msg.setSentDate(new Date());

msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to, false));

Transport.send(msg);

} catch (UnsupportedEncodingException | MessagingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

JOptionPane.showMessageDialog(null, "Message send");

}

public void setProperties(Properties mailProps) {

this.props = mailProps;

}

public void setBody(String body) {

this.body = body;

}

public void setSubject(String subject) {

this.subject = subject;

}

public void setParam(String user, String pass, String to) {

this.from = user;

this.password = pass;

this.to = to;

}

}

package ua.cn.stu.lab7;

import java.util.Properties;

import javax.mail.Folder;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.NoSuchProviderException;

import javax.mail.Session;

import javax.mail.Store;

import javax.swing.JTextArea;

public class MailChecker extends Thread {

private volatile boolean status = true;

private Properties props;

private String user;

private String password;

private JTextArea textArea;

private double msgCount = 0;

public MailChecker() {

super();

}

public void setParam(String user, String pass, JTextArea textArea) {

this.user = user;

this.password = pass;

this.textArea = textArea;

}

@Override

public void run() {

if (props == null) {

throw new NullPointerException("mail properties not initialaized");

}

if (user == null || password == null) {

throw new NullPointerException("mail properties have null value");

}

int delay = 10000;

while (status) {

try {

props.put("mail.smtp.user", user);

props.put("mail.smtp.password", password);

Session emailSession = Session.getDefaultInstance(props);

Store store = emailSession.getStore("pop3s");

store.connect("pop.gmail.com", user, password);

// create the folder object and open it

Folder emailFolder = store.getFolder("INBOX");

emailFolder.open(Folder.READ\_ONLY);

// retrieve the messages from the folder in an array and print

// it

Message[] messages = emailFolder.getMessages();

textArea.append("\nmessages.length - " + messages.length);

if (msgCount != messages.length) {

int i = 0;

for (Message currMessage : messages) {

textArea.append("\n---------------------------------");

textArea.append("\nEmail Number " + (i + 1));

textArea.append("\nDate: " + currMessage.getSentDate());

textArea.append("\nSubject: " + currMessage.getSubject());

textArea.append("\nFrom: " + currMessage.getFrom()[0]);

// textArea.append("\nText: \n" +

// currMessage.getContent().toString());

textArea.append("\n---------------------------------");

++i;

}

msgCount = messages.length;

}

// close the store and folder objects

emailFolder.close(false);

store.close();

} catch (NoSuchProviderException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

try {

sleep(delay);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public void setStatus(boolean status) {

this.status = status;

}

public void setProperties(Properties properties) {

this.props = properties;

}

}

**7.3** **Выводы**

В ходе лабораторной работы было создано приложение с использованием Java Mail API, которое производит чтение всех писем из почтового ящика пользователя. Также с помощью данной программы можно отправлять письма по указанному адресу электронной почты.